9H

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/754,017	01/07/2004	Kenneth W. Cowan	NUM-02402	7115

26339 7590 12/20/2006
MUIRHEAD AND SATURNELLI, LLC
200 FRIBERG PARKWAY, SUITE 1001
WESTBOROUGH, MA 01581

EXAMINER	
VU, TUAN A	
ART UNIT	**PAPER NUMBER**
2193	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	12/20/2006	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No.	Applicant(s)
	10/754,017	COWAN ET AL.
	Examiner	Art Unit
	Tuan A. Vu	2193

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 1/07/04.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-42 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-42 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on 07 January 2004 is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.	6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the application filed 1/07/2004

Claims 1-42 have been submitted for examination.

Double Patenting

2. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory obviousness-type double patenting rejection is appropriate where the conflicting claims are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g., *In re Berg*, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent either is shown to be commonly owned with this application, or claims an invention made as a result of activities undertaken within the scope of a joint research agreement.

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

3. Claims 1, 22, 20 and 41 are rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claims 8, 35 of U.S. Patent No. 6,701,519 (hereinafter '519). Although the conflicting claims are not identical, they are not patentably distinct from each other because of the following.

As per **instant claims 1, 22**, '519 claims 8, and 35 recite

collection of *records of platform information, code coverage data* for *programs prior to their execution* -- thus this is equivalent to extracting build information of builds using compilation process, with *record of platform information* equivalent to registering *build information* for a software program code or product and its registered information (refer to '518 *component* and system *configuration data* information on product, thus a repository of configuration record);

records including *software component* and *configuration data* including a module name, a *file version, product version* -- this is equivalent to determining at runtime of software program the version thereof being tested;

performing intersection of *sets of records* to form a *resulting set* corresponding to the above collected records of platform data based on *test coverage results* and *platform information* being matched via criteria -- this is equivalent to determining a first one builds corresponding to runtime software module information.

As per **claims 20 and 41**, the '519 claims 8 and 35 also extracting build information of build of a compiled program prior to execution; a registered configuration data recorded for the build information or plurality of previous records; executing the software to collect coverage results; apply criteria to map coverage data to platform information; and determining testing via making a intersection with sets of records determined from the criteria assessing using records of information retrieved from such platform register or software configuration data.

Claim Rejections - 35 USC § 101

4. 35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

5. Claims 1-2, 4, 6-23, 25, 27-42 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

> The Federal Circuit has recently applied the practical application test in determining whether the claimed subject matter is statutory under 35 U.S.C. § 101. The practical application test requires that a " useful, concrete, and tangible result" be accomplished. An "abstract idea" when practically applied is eligible for a patent. As a consequence, an invention, which is eligible for patenting under 35 U.S.C. § 101, is in the "useful arts" when it is a machine, manufacture, process or composition of matter, which produces a concrete, tangible, and useful result. The test for practical application is thus to determine whether the claimed invention produces a "useful, concrete and tangible result".

Specifically, claims 1 and 22 recite a method or computer product for registering builds and determining build information about version and determining a build corresponding to said information. The mere fact of determining by computer software instructions remains a internal decision that cannot be construed as externalized as real world application results as required by the Practical Application Test; the claims as a whole amount to a abstract non-practical application, i.e. lacking teaching reasonably conveying that a concrete, useful tangible result is generated (based on a determination state). The claims are rejected for leading to a non-statutory subject matter.

Claims 2, 4, 6-15, 23, 25, 27-36 are also rejected for not remedying to the above deficiency.

Claims 16 and 37 also recites determining based on build information using metrics on code change. As set forth above, the mere step of determining amounts to an internal abstract software decision that remains internal, not a real-world tangible and concrete result in terms of application level useful data or entities. The claims are rejected for leading to non-statutory subject matter.

Claims 17-19, 38-40 for failing to remedy the above deficiency are equally rejected.

Claims 20-21, 41-42 for amounting to a internal abstract concept (determining step) as a whole, not fulfilling the Practical Application Test requirement, are also rejected as non-statutory subject matter.

Claim Rejections - 35 USC § 112

6. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

7. Claims 20-21, 41-42 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

8. Claims 20, 41 present the limitation recited as 'determining testing associated with a build ...' does not seem to be constructed in a manner that would define the metes and bounds of what actually is being determined. The term 'testing' is a vague way of describing an ongoing action, and *determining* entails a precise scope of information; and when put together, the combination 'determining testing' does not convey what exactly this *determining* or *testing* is all about. One skill in the art would not be able to construe what 'determining testing' amounts to because *testing* requires action taken within some implementation settings; thus, *testing* will be treated as any form of test-related information. Claims 21 and 42 are also rejected, and appropriate correction is required.

Claim Rejections - 35 USC § 102

9. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

10. Claims 1-10, 16-31, 37-42 are rejected under 35 U.S.C. 102(b) as being anticipated by Leblang et al., USPN: 5,574,898 (hereinafter Leblang).

As per claim 1, Leblang discloses a method for automatically tracking build information comprising:

extracting build information (derived objects, configuration record – col. 25, lines 27-55) from one or more builds wherein said one or more builds are produced using a compilation (see *foo.c, foo.h, foo.o* - Fig. 23; Fig. 20) process;

registering said one or more builds by storing said build information (e.g. VOB – Figs. 6-7; repository 102, VOB – Fig. 24) corresponding to each of said one or more builds;

automatically determining a runtime software module information about a version of software being tested (e.g. col. 9, lines 10-55; *fixing a bug, test it* – col. 9, lines 40-66 – Note: check-out – see Fig. 9, 14 -- from version control system to check version delta – see Fig. 12-13 -- for developing code reads on determine runtime module information for module being tested); and

automatically determining a first of said one or more builds corresponding to said runtime software module information (e.g. col. 10, lines 44 to col. 11, line 57; Fig. 7; Fig. 22-24; col. 13, line 41 to col. 14, line 47 – Note: using configuration rules – see col. 9, lines 10-55 – to check appropriate version of components to be enlisted for a project of fixing a bug reads on determining a build that correspond to the runtime information, runtime being the enlistment of project derived files or objects).

As per claim 2, Leblang discloses:

storing build information in a database; and using said build information in said database to automatically determine said first build corresponding to said runtime software module information.

As per claim 3, Leblang discloses wherein said database that includes said build information is an object database (VOB – col. 9, lines 40-66; Fig. 24;), and creating and storing one or more objects corresponding to each of said builds; and creating and storing one or more objects corresponding to software modules included in each of the builds (Fig. 17, 20-21; 23, 24).

As per claim 4, Leblang discloses including:

creating and storing a session object (e.g. view: alpha –Fig. 21, 22; view … dynamic, can be modified - col. 9, lines 25-44 – Note: host client machine opening one instance of view reads on session object – see col. 12, lines 12-15, 49-53) corresponding to a test session (a check out of components for a release as to fixing a bug reads on test session – see col. 9, line 45 to col. 10, line 24) of said version of software;

creating and storing one or more objects (Fig. 20-21; 23-24; col. 10, lines 43-51) corresponding to software modules describing said runtime software module information;

automatically determining a previously created build object (VOB- Fig. 23-24; VOB – Fig. 6) corresponding to one of said one or more builds previously registered; and storing an address of said previously created build object in said session object (e.g. entries *532* , record *102* – Fig. 23; Fig. 17; Fig. 8; */usr/hw/src/msg.o@@3-May.08:12* – Fig. 22).

As per claims 5-6, Leblang discloses storing software module information about at least one module for each of said one or more builds (e.g. Fig. 23-24); wherein said automatically determining at runtime software module information about a version of software being tested further includes gathering runtime information (e.g. col. 15 line 64 to col. 17, line 61 – Note: developer retrieving objects based on metadata during enlistment of persisted build components reads on gathering at runtime of development session) about software modules dynamically loaded (Note: view per developer reads on dynamic loading of code files – see Fig. 6; col. 9, line 8 to col. 10, line 35) in said computer system.

As per claim 7, Leblang discloses subroutine calls (e.g. *checkout, checkin* - col. 18, lines 29-33; *Operating System/File system* 552 – Fig. 23) associated with an operating system executing in said computer system are used in said gathering runtime information.

As per claim 8, Leblang discloses using said build information in said database to automatically determine a second of said one or more builds (e.g. Fig. 17; Fig. 23)corresponding to software module information included in a bug report (e.g. *fixing a bug, test it* – col. 9, lines 40-66); and

creating and storing a bug report object corresponding to said bug report (Note: VOB record of derived object per access from a view for developing a release – see col. 13, line 41 to col. 14, line 47-- addressing a bug reads on bug report being store), said bug report object including an address associated with said second build (entries 532 - Fig. 22; File system -Fig. 23-24).

As per claim 9, Leblang discloses submitting said bug report included in a formatted electronic message (col. 13, lines 17-29 – NOTE: a file system call reads on electronic message); interpreting data included in said formatted electronic message (*checkout, checkin* -

col. 18, lines 29-33; *Operating System/File system* 552 – Fig. 23) to enable said determining of said second build.

As per claim 10, Leblang discloses

creating and storing another build object corresponding to a build in which said bug report is identified as being corrected; and associating said other build object with said bug report object in said database (Fig. 23-24 – Note: release and VOB of previous builds reads on bug report of previous fix being persisted for reuse).

As per claim 16, Leblang discloses a method for determining code volatility, the method comprising:

extracting build information for at least two builds (e.g. derived objects, configuration record – col. 25, lines 27-55) wherein said at least two builds are produced using a compilation process (see *foo.c, foo.h, foo.o* - Fig. 23; Fig. 20);

registering said at least two builds by storing said build information corresponding to each of said at least two builds (e.g. repository 102, VOB – Fig. 24);

identifying a first and a second of said at least two builds; determining code volatility using said build information about said first and said second builds, said code volatility being determined using one or more metrics (e.g. Fig. 12-13, 15-16) representing an amount of code change that has occurred between said first build and said second build.

As per claim 17, Leblang discloses determining a total number of functions of said first build and said second build; determining a percentage of functions added in accordance with said total number of functions; determining a percentage of functions removed in accordance with said total number of functions (e.g. Fig. 12-13, 15-16 – Note: code being removed, added in *delta* record reads on a relative portion – or percent of source code functions – of the ancestor version as set forth in the tree of stored versions) ;

determining a percentage of functions modified in accordance with said total number of functions; and

determining code volatility as a sum of said percentage of functions added, said percentage of functions removed, and said percentage of functions modified (Note: sum of percent or portion of code being recorded as delta reads on code volatility computed as sum of source code functions changes).

As per claim 18, Leblang discloses wherein said build information is stored in a database (VOB – Figs. 6-7; repository 102, VOB – Fig. 24; col. 10, line 36-51) and used in determining said code volatility.

As per claim 19, Leblang discloses determining differences in a function in which a first version of a function is associated with one module and a second version of the function is associated with a second module, said first module being associated with a first build, and said second module being associated with a second build (e.g. Clearmake col. 15, lines 37 col. 16, line 16); and using checksum or function signature information (e.g. *checksum* - col. 31, lines 5-38) for determining differences.

As per claim 20, Leblang discloses a method for tracking build information comprising:

extracting build information (e.g. derived objects, configuration record – col. 25, lines 27-55) from one or more builds wherein said one or more builds are produced using a compilation process (see *foo.c, foo.h, foo.o* - Fig. 23; Fig. 20);

registering said one or more builds by storing said build information in a database (e.g. repository 102, VOB – Fig. 24);

testing software (e.g. *fixing a bug, test it* – col. 9, lines 40-66) that includes one or more modules;

automatically determining a matching build (e.g. col. 10, lines 44 to col. 11, line 57; Fig. 7; Fig. 22-24) for said one or more modules associated with one of said builds previously registered; and

determining testing associated with a build by performing a query (e.g. *audit entries* – Fig. 23) using said information in said database (Note: generating a audit record reads on performing a query from database of configuration previously stored).

As per claim 21, Leblang discloses determining code volatility (e.g. Fig. 12-13, 15-16) between two predetermined builds having corresponding build information stored in said database.

As per claim 22, Leblang discloses a computer program product for automatically tracking build information comprising machine executable code for:

extracting build information from one or more builds wherein said one or more builds are produced using a compilation process;

registering said one or more builds by storing said build information corresponding to each of said one or more builds;

determining at runtime software module information about a version of software being tested; and for determining a first of said one or more builds corresponding to said runtime software module information;

all of which action steps having been addressed correspondingly in claim 1.

As per claims 23-25, 29-31, these claims correspond to the subject matter of claims 2-4, 8-10, respectively; hence will be rejected according to the rationale set forth therein, respectively.

As per claims 26, and 27-28, these claims correspond to the subject matter of claims 5, and 6-7, respectively; hence will be rejected according to the rationale set forth therein, respectively.

As per claims 37-40, these claims correspond to the subject matter of claims 16-19, respectively; hence will be rejected according to the rationale set forth therein, respectively.

As per claim 41, Leblang discloses a computer program product for tracking build information comprising machine executable code for:

extracting build information from one or more builds wherein said one or more builds are produced using a compilation process; registering said one or more builds by storing said build information in a database; for testing software that includes one or more modules; for automatically determining a matching build for said one or more modules associated with one of said builds previously registered; and for determining testing associated with a build by performing a query using said information in said database;

all of which step limitations having been addressed in claim 20.

As per claim 42, refer to rejection of claim 21.

Claim Rejections - 35 USC § 103

11. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

12. Claims 11-15, 32-36 are rejected under 35 U.S.C. 103(a) as being un-patentable under Leblang et al., USPN: 5,574,898.

As per claim 11, Leblang discloses wherein said automatically determining said first build further comprises:

determining said first build for which a matching build is being determined from said one or more builds registered; determining a candidate list including one or more builds having at least one module in common with said first build (e.g. Fig. 23-24; col. 13, line 41 to col. 14, line 47); for each build included in said candidate list, determining if modules included in said each build match modules included in said first build (e.g. verify versions Fig. 24; col. 20, line 45 to col. 21, lines 63 – Note: each set of checkout files for a build reads on a candidate list);

But Leblang does not explicitly disclose

for each build included in said candidate list, if there are no modules included in said each build that have a module name and associated attributes matching a module included in said first build, determining that said each build is not a match for said first build; and

for each build included in said candidate list, if a first of said modules included in said each build has a module name that matches a module included in said first build but attributes associated with said module do not match said module included in said first build, determining that said each build is not a match for said first build.

Based on name and attributes associated with a record of previous release (see Fig. 1, 6,-7, 9, 17, 20-24) whereby derived objects are enlisted for a session of developer's view based on version control system, the system routines for matching version name and attributes per file stored in the VOB or check-out control system are intrinsic to the above user's actions col. 20, line 45 to col. 21, lines 63), the matching of name and attributes to see if versions stored correspond to a particular release is strongly suggested. It would have been obvious for one skill in the art at the time the invention was made to implement the checkout and user's version verifying in the tree-based integration of proper files so that for each build in said candidate list as incrementally added to the tree, so that name of build (see Fig. 17) as well as attributes (see Fig. 9; Fig. 23) are also checked and verified (see audit entries – Fig. 23) to assess whether such version of build components (as candidate list) would still be appropriate to be included in the tree and enlistment view for the bug fix project (see col. 13, line 41 to col. 14, line 47; col. 9, line 8 to col. 10, line 35) .

As per claim 12, Leblang discloses further including:

for each build included in said candidate list, determining a number of matches for modules included in said each build having a matching module name and attributes of a module included in said first build (see rationale as set forth in claim 11 – Note: each set of files or build component per check out reads on list of candidate builds);

determining a valid list of one or more matching builds, each of said builds included in said valid list having a full match for modules included in said each build with modules included in said first build, each module included in said each build having a corresponding matching module included in said first build, said name and associated attributes of said each module matching said matching module included in said first build (see Fig. 23-24; col. 20, line 45 to col. 21, lines 63 – Note: the result of assessing the entries based on references from the version checking system and the configuration record leading to derived objects reads on valid list as a result from checking on checkout builds or candidate list);

But Leblang does not explicitly disclose

determining a maybe list of one or more builds, each of said one or more builds included in said maybe list having at least one module having a module name that does not have a matching module included in said first build, said each build including at least one module having a module name and associated attributes matching another module included in said first build, wherein said each build has an associated number of matches.

But based on the candidate nature of the checkout build components being enlisted and checked as set forth above in claim 11, the above name and attribute mapping and determining steps would have been obvious as set forth in the rationale of claim 11.

As per claim 13, Leblang discloses

if there is only one build included in said valid list, determining that said one build matches said first build; and if there is more than one build included in said valid list, selecting one of the more than one builds included in said valid list as being the build matching said first build (see claim 11 - Fig. 23-24; col. 13, line 41 to col. 14, line 47); performing an alternative action if said valid list is empty (Note: when a release is targeted for a rebuild, new fix associated with an starting empty list reads on alternative action if the fix release for target is not founded on any legacy of record -- see col. 10, lines 59-61).

However, Leblang does not explicitly disclose if said valid list is empty, for each project, adding a build from said maybe list to said valid list in which the build added has a maximum number of matches of builds associated with said each project, a project having one or more associated builds. But the maybe list has been addressed above in claim 12, hence would have been obvious herein in regard to the rationale set forth therein.

As per claims 14-15, Leblang discloses selecting one or more build associated with a software project (e.g. version selector – col. 2, line 54 to col. 3, line 19); determining a matching build in accordance with a predetermined build selected from a list of more than one build previously registered (e.g. Fig. 9, 17, 20).

As per claims 32-36, these claims correspond to the subject matter of claims 11-15, respectively; hence will be rejected according to the rationale set forth therein, respectively.

Conclusion

13. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (272) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on (571)272-3756.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

[illegible]

Tuan A Vu
Patent Examiner,
Art Unit 2193
December 18, 2006